ブロックの作成

沼田　賢一

富山県立大学　情報基盤工学講座

1. はじめに
2. 提案手法
3. おわりに

August 27, 2020

はじめに

本研究の背景

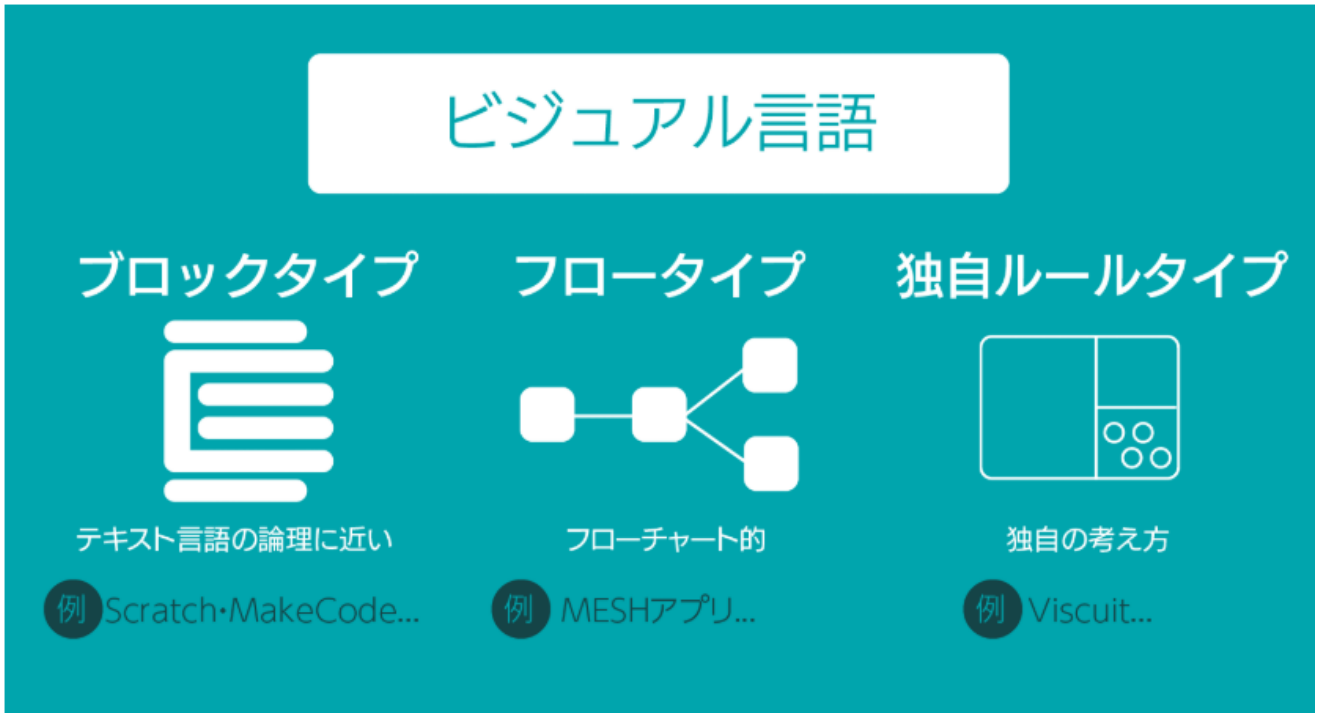
近年，企業などでは世間に溢れる様々な情報を収集し，ビックデータと呼ばれる非常に巨大で複雑なデータの集合として扱うことが増えてきている．また，ビックデータを扱うには様々な処理や解析によって情報を取捨選択し，自分たちに必要な形で保管する必要がある．しかし，ビックデータの情報量は膨大で，人の手で全て解析するのは困難であるため，一般にプログラミング言語を用いて機械に処理させるのが一般的である．このため，ビックデータを扱うためにはプログラミングの知識や技術が必要不可欠であり，プログラミングに触れたことがない人には扱いづらいものになっている．

目的

プログラミング初心者でも扱いやすいビジュアルプログラミング言語を使い，ブロック１つ１つに処理を対応させることでブロックを並べるだけで処理できるプログラムを作成することができるようにする．プログラミングができない人でもビックデータを扱うことができるようにする．

ビジュアルプログラミング言語

プログラムをテキストで記述するのではなく，視覚的なオブジェクトで記述するプログラミング言語のこと．視覚的でわかりやすいものが多いため，プログラムの組み立て方を学ぶのに有効であると注目されている．



ブロックタイプの VPL

ブロックタイプの VPL

機械学習（人工知能・AI）を使って課題を解決するクラウドサービスの MAGELLAN BLOCKS（BLOCKS）や教育用作られ様々なアプリケーションに応用して使われている Blockly などがある．
応用して使われているサービスとして Scratch や MakeCode が存在する．

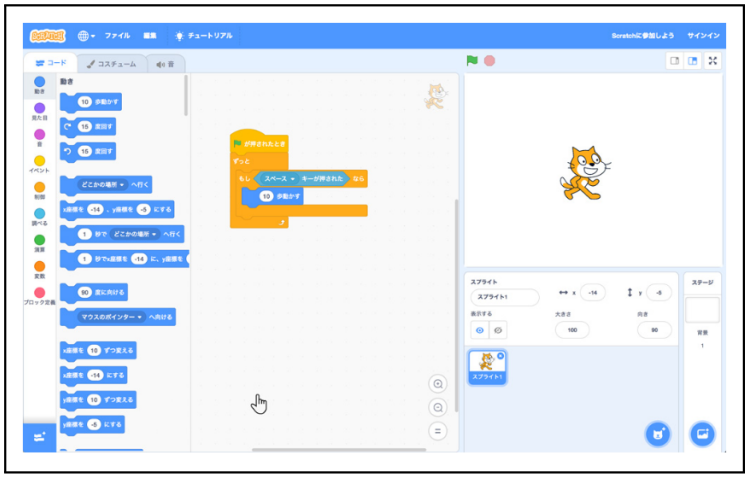
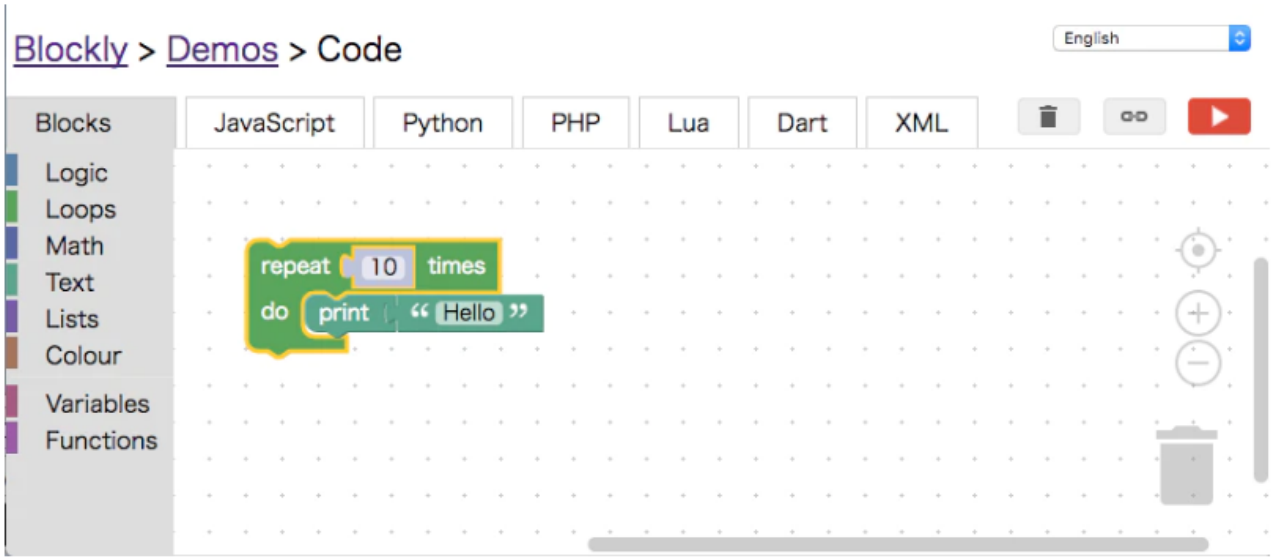


Figure: scratch

Figure: makecode

Blockly

Google が提供しているビジュアルプログラミング言語のライブラリ．簡単な記述で自分だけのビジュアルプログラミング言語を作ることができる．
また，作成したブロックは JavaScript や python,PHP,Lua,Dart などのプログラミング言語にエキスポートすることができる．

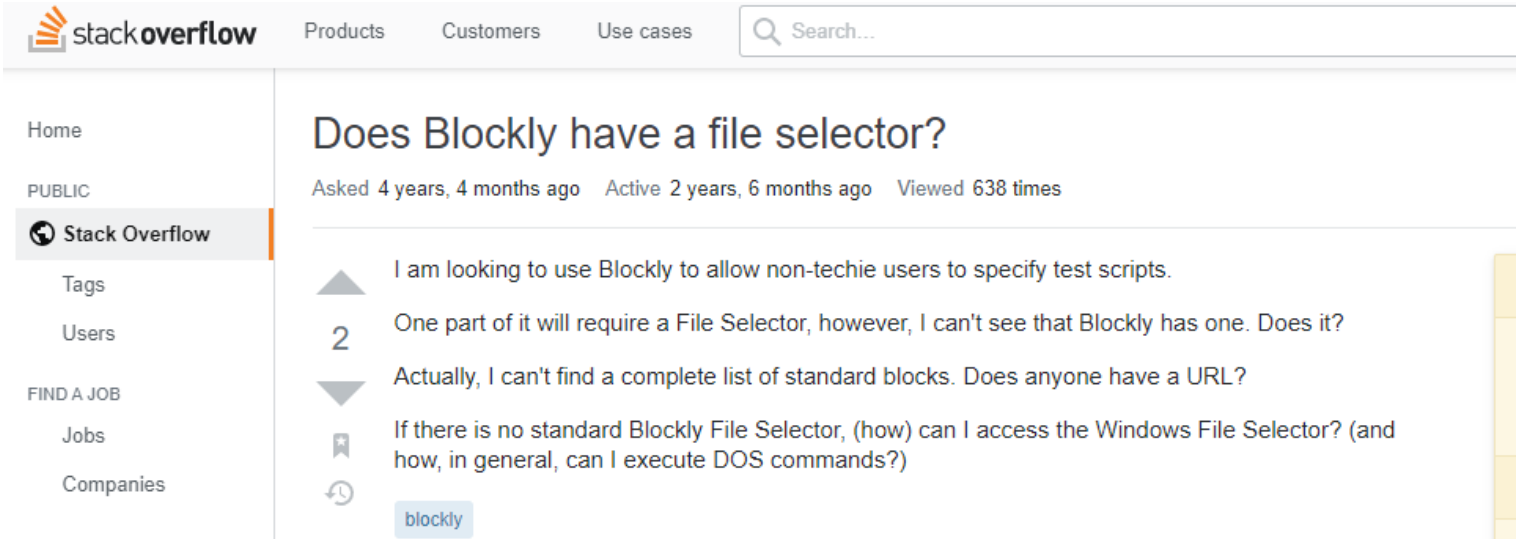


Blockly の問題点

ビックデータを扱うとき，外部からデータ (CSV) を読み込む必要があるが blockly のデモコードには外部からのファイルの読み込みが実装されていない．

CSV の読み込み方

デモコードでは，実装されていないが Blockly をもとに作られている Scratch では CSV の読み込みが実装されている．また，stackoverflow で読み込み方を質問していたので参考にして作る．



カスタムブロック

カスタムブロックとは

Blockly では，もともとあるブロックの他にユーザが好きなブロックを作成することができる．
CSV ファイルを読み込むためには，カスタムブロックを自作する必要がある．

カスタムブロックを作る前に..

js でローカルファイルを読み込む方法は，調べると色々出てくる．
しかし，カスタムブロックに実装する方法はない．カスタムブロックの構成を理解していないと簡単には導入できない．

カスタムブロック

カスタムブロックの構成

1. ブロックの定義
2. コードの生成
3. ブロックのカテゴリーと配置決め

Blockly Developer Tools

簡単にカスタムブロックを作成する支援ツールとして，Blockly Developer Tools がある．
この支援ツールは blockly を用いて，ブロックを作ることができる．

カスタムブロック

1. ブロックの定義

作成したいブロックの外観とブロックに接続する数値やテキストをここで定義する.
外観は，ブロックの色やブロックの接続 (構文ブロックと値ブロック)，表示する文字等がある.
また，ブロック内の空きに何を入力 (input)or 出力 (output) とするか決める.

Figure: block の種類

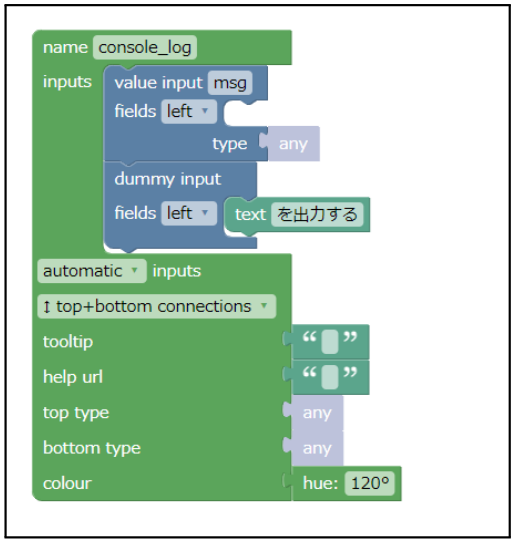


Figure: consol log に結果を出力する関数

カスタムブロック

2. コードの生成

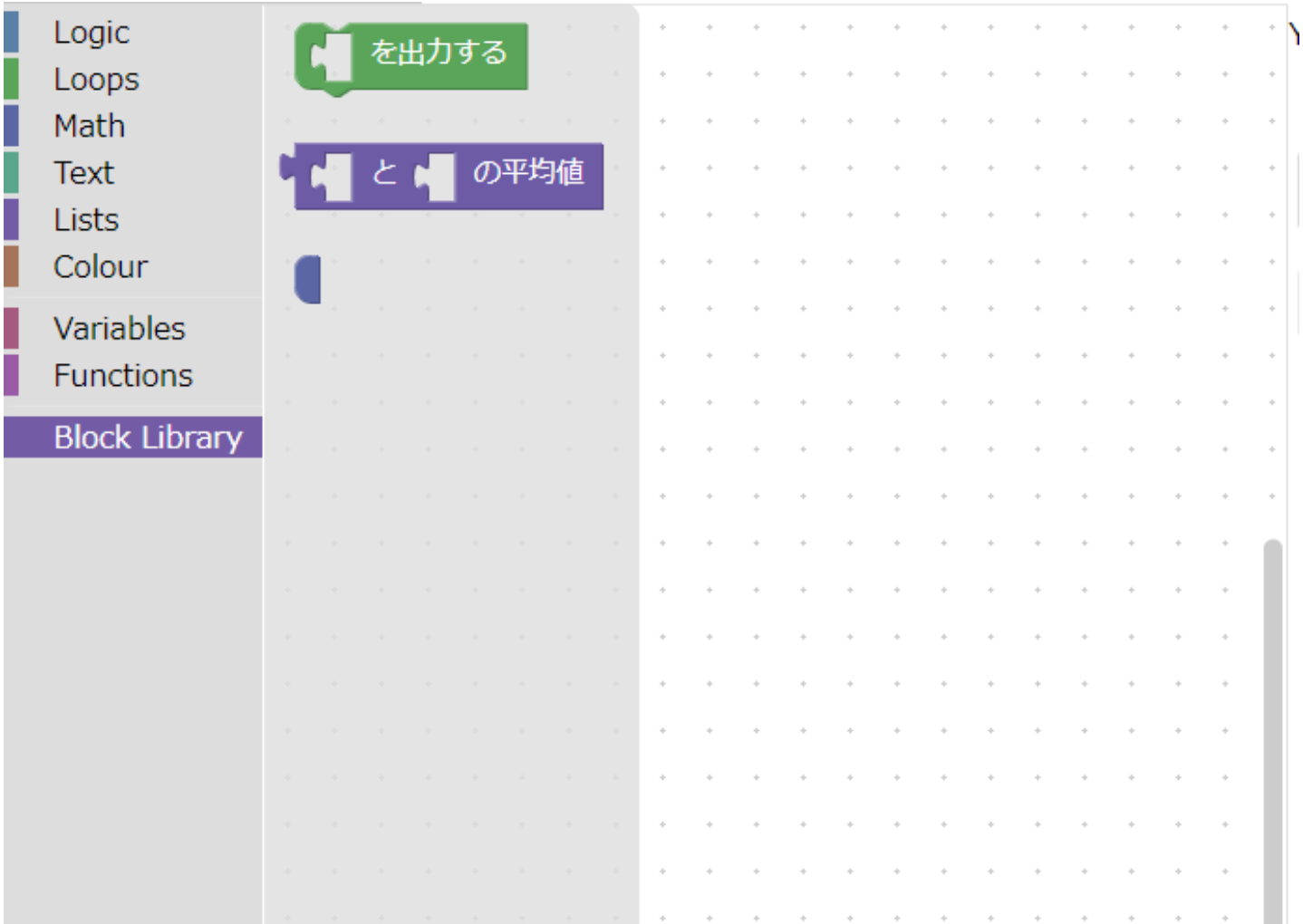
コードの生成では，ブロックの動作の定義を行う．例えば，平均値を出すブロックを作成するときは，平均を出す計算部分をここで書く．

```
erage.js > ...
Blockly.JavaScript['average'] = function(block) {
  var value_v1 = Blockly.JavaScript.valueToCode(block, 'v1', Blockly.JavaScript.ORDER_ATOMIC);
  var value_v2 = Blockly.JavaScript.valueToCode(block, 'v2', Blockly.JavaScript.ORDER_ATOMIC);
  // TODO: Assemble JavaScript into code variable.
  var code = '(' + value_v1 + '+' + value_v2 + ')/2';
  // TODO: Change ORDER_NONE to the correct strength.
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

カスタムブロック

3. ブロックのカテゴリーと配置決め

作ったブロックをどこのカテゴリーに入れるかを決める．



進捗

今回の目標

・前回，csv を読み込むポップアップ画面を出せるようにしたブロックを編集して，csv の値をブラウザ上で一時保存できるようする．
・回帰分析のブロックを作り，キャンバスに結果を可視化できるようにする．

進捗

進捗状況

まず，CSV ファイルの読み込みブロックについての進捗.
うまくファイルを読み込無事ができるようになった.
問題点としては，読み込みと同時に配列に変換したいがなぜか，改行が邪魔をしてきれいな配列に落とし込むことができていない点.
実演

```
Blockly.JavaScript['import file'] = function(block) {
  var value = block.getFieldValue("value_csv", Blockly.JavaScript.ORDER_ATOMIC);
  value = value.split('\n');
  //value=value.replace(/\r?\n/g, '')
  //value =value[0].split(',');
  //var val1 =new Array([[1,2],[15,5]])
  //var value1 = [[1,2],[15,5]]
  //var st=string(value);
 // var value=value.replace('\n', '');

/*   var csv_value =value.split('');
  var csvArray=[]
  for(let i = 0; i < csv_value.length; i++) {
    csvArray[i]=csv_value[i]
  }
  var csvArray1=value.length */
  //value=String(value)
  //var csv_value =value.split('\n');
  //var csvArray=new Array()
  //csvArray[0]=value[0]
```

進捗状況

```
Blockly.JavaScript['import file'] = function(block) {
  var value = block.getFieldValue("value_csv", Blockly.JavaScript.ORDER_ATOMIC);
  value = value.split('\n');
  //value=value.replace(/\r?\n/g, '')
  //value =value[0].split(',');
  //var val1 =new Array([[1,2],[15,5]])
  //var value1 = [[1,2],[15,5]]
  //var st=string(value);
 // var value=value.replace('\n', '');

/*   var csv_value =value.split('');
  var csvArray=[]
  for(let i = 0; i < csv_value.length; i++) {
    csvArray[i]=csv_value[i]
  }
  var csvArray1=value.length */
  //value=String(value)
  //var csv_value =value.split('\n');
  //var csvArray=new Array()
  //csvArray[0]=value[0]
  //csvArray[1]=value[2]

  /* var code = value.newBlock("lists_create_with_container");
  code.initSvg(); */

  //var code = 'console.log(typeof(' +value +'))'
  var code = '[' + value[1]+ ']';
  //var code = value1;
  return [code,Blockly.JavaScript.ORDER_MEMBER];
  //return typeof(csvArray);
};
```

上の図は作成したブロックの動作のプログラム

進捗状況

```
Blockly.Blocks['import file'] = {
  init: function() {
    var fileInput = new Blockly.FieldTextInput('** ファイルを選択してください **');
    let block = this.id;
    this.appendDummyInput()
        .appendField(fileInput, "value_csv");
        fileInput.showEditor_=(()=>{
          var input = document.createElement('input');
          input.type = 'file';
          input.accept = '.csv';

          input.onchange = e => {
            var file = e.target.files[0];
            var reader = new FileReader();
            reader.readAsText(file,'UTF-8');
            reader.onload = readerEvent => {
              var content = readerEvent.target.result;
              //var contents=content.split("\n");
             // var contents = [[1,2],[15,5]]
              console.log(content);
              //console.log(typeof(content));
              //console.log(contents);
              //console.log(typeof(contents));
              //console.log(block);
              let blockyy = Blockly.mainWorkspace.getBlockById(block);
              blockyy.setFieldValue(content, "value_csv");
            }
          }
          input.click();
        }
        )
    this.appendDummyInput()
        .appendField("を読み込み");
    this.setInputsInline(true);
    this.setOutput(true,"Array");
    this.setColour(285);
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

上の図は作成したブロックの見た目に関するプログラム

進捗

進捗状況

回帰分析のブロックを作り，キャンバスに結果を可視化できるようにすることの進捗について．
回帰分析を求めるプログラムは，時間がなかったので読み込むブロックとセットのブロックになってしまっているが，一応完成した．
このあと，配列問題が解決したら，csv 読み込むブロックと回帰分析の求めるブロックを分割する必要がある．
実演

おわりに

今回の進捗

1. 前回ポップアップで終わってしまっていた csv 読み込みブロックが完成した
2. 回帰分析を求めることができるようになった.

今後の課題

1. 読み込み後の，配列をきれいにする.
2. 配列をきれいにできたら，回帰分析のプログラムを分割する.
3. キャンバスに回帰分析の結果を描画できるようにする.